UNITED STATES PATENT APPLICATION

FOR

Method and Apparatus for Forwarding Requests in a Cache Hierarchy Based on User-Defined Forwarding Rules

INVENTOR:

J Eric Mowat

Prepared by:

Blakely, Sokoloff, Taylor & Zafman LLP
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025
(408) 720-8300

Attorney's Docket No. 5693P005
(Client Docket No. P01-1296)

Method and Apparatus for Forwarding Requests in a Cache Hierarchy Based on User-Defined Forwarding Rules

FIELD OF THE INVENTION

[0001] The present invention pertains to devices which proxy requests between clients and servers and which cache content on a computer network. More particularly, the present invention relates to a method and apparatus for forwarding requests in a cache hierarchy based on user-defined forwarding rules.

BACKGROUND OF THE INVENTION

[0002] Of the many uses of the Internet, one of the more common ones is to access content on a remote server, such as a World Wide Web server. Typically, a person operates a client device to access content on a remote origin server over the Internet. The client may be, for example, a personal computer (PC) or a handheld device such as a personal digital assistant (PDA) or cellular telephone. The client often includes a software application known as a browser, which can provide this functionality. A person using the client typically operates the browser to locate and select content stored on the origin server, such as a web page or a multimedia file. In response to this user input, the browser sends a request for the content over the Internet to the origin server on which the content resides. In response, the origin server returns a response containing the requested content to the client, which outputs the content in the appropriate manner (e.g., it displays the web page or plays the audio file). The request and

response may be communicated using well-known protocols, such as transmission control protocol/Internet protocol (TCP/IP) and hypertext transfer protocol (HTTP).

[0003] For a variety of reasons, it may be desirable to place a device known as a proxy logically between the client and the origin server. For example, organizations often use a proxy to provide a barrier between clients on their local area networks (LANs) and external sites on the Internet by presenting only a single network address to the external sites for all clients. A proxy normally forwards requests it receives from clients to the applicable origin server and forwards responses it receives from origin servers to the appropriate client. A proxy may provide authentication, authorization and/or accounting (AAA) operations to allow the organization to control and monitor clients' access to content. A proxy may also act as (or facilitate the use of) a firewall to prevent unauthorized access to clients by parties outside the LAN. Proxies are often used in this manner by corporations when, for example, a corporation wishes to control and restrict access by its employees to content on the Internet and to restrict access by outsiders to its internal corporate network. This mode of using a proxy is sometimes called "forward proxying".

[0004] It is also common for a proxy to operate as a cache of content that resides on origin servers; such a device may be referred to as a "proxy cache". An example of such a device is the NetCache product designed and manufactured by Network Appliance, Inc. of Sunnyvale, California. The main purpose of

caching content is to reduce the latency associated with servicing content requests. By caching certain content locally, the proxy cache avoids the necessity of having to forward every content request over the network to the corresponding origin server and having to wait for a response. Instead, if the proxy cache receives a request for content which it has cached, it simply provides the requested content to the requesting client (subject to any required authentication and/or authorization) without involving the origin server.

[0005] Proxy caches may be used by corporations and other institutions in the forward proxying mode, as described above. Proxy caches are also commonly used by high-volume content providers to facilitate distribution of content from their origin servers to users in different countries or other geographic regions. This scenario is sometimes called "reverse proxying". As an example of reverse proxying, a content provider may maintain proxy caches in various different countries to speed up access to its content by users in those countries and to allow users in different countries to receive content in their native languages. In that scenario the content provider "pushes" content from its origin servers to its proxy caches, from which content is provided to clients upon request.

[0006] Often a proxy cache is one of a number of proxy caches distributed on the Internet in a defined logical hierarchy known as a "cache hierarchy". Typically, each proxy cache has knowledge of the other proxy caches in the cache hierarchy. Consequently, when a proxy cache receives a content request but does not have the requested content cached locally (a "cache miss"), it may

forward the request to another proxy cache in the hierarchy according to a predefined forwarding scheme. Assuming a proxy cache in the hierarchy has the requested content, the latency in servicing the request generally will still be lower than if the request had to be passed all the way to the origin server.

[0007] One problem with proxy caches in the known prior art is that they provide only crude control over how requests are forwarded. A proxy cache according to the known prior art typically uses a small number of (e.g., four or five) control variables to control request forwarding. These control variables interact with each other in complex, non-intuitive ways, making it very difficult for users (e.g., network administrators) to control the manner in which the proxy cache forwards requests. As a result, there tends to be an undesirably high number of support calls to the vendor of the proxy cache when users are unable to properly configure their devices. These prior art proxy caches also provide no ability to control the sequence in which forwarding rules are applied to a request. In addition, such devices make it difficult to add functionality after deployment, because doing so normally requires adding one or more control variables, which must work in conjunction with the previously-programmed control variables. Making such a change causes testing and validation of the proxy cache to become more difficult and complex. What is needed, therefore, is a technique which overcomes these disadvantages of the prior art.

SUMMARY OF THE INVENTION

[0008] The present invention includes a method and apparatus for operating an intermediary node on a network. The method comprises receiving, at the intermediary network node, a request for content on a network, and determining, in the intermediary network node, a forwarding destination in a defined forwarding hierarchy, by applying a set of user-specified forwarding rules to the request. The request is then forwarded according to the determined forwarding destination.

[0009] Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follows.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like references indicate similar elements and in which:

[0011] Figure 1 illustrates a network environment in which a proxy cache according to the present invention may be implemented;

[0012] Figure 2 illustrates a block diagram of the components of the proxy cache, according to one embodiment;

[0013] Figure 3 shows examples of display screens which may be generated by the user interface to allow a user to specify forwarding rules;

[0014] Figure 4 is a flow diagram showing a request forwarding process that may be executed in the proxy cache, according to one embodiment; and

[0015] Figure 5 is block diagram showing an abstraction of the hardware components of the proxy cache, according to one embodiment.

DETAILED DESCRIPTION

[0016] A method and apparatus for forwarding requests in a cache hierarchy based on user-defined forwarding rules are described. Note that in this description, references to "one embodiment" or "an embodiment" mean that the feature being referred to is included in at least one embodiment of the present invention. Further, separate references to "one embodiment" in this description do not necessarily refer to the same embodiment; however, neither are such embodiments mutually exclusive, unless so stated and except as will be readily apparent to those skilled in the art. For example, a feature, structure, act, etc. described in one embodiment may also be included in other embodiments. Thus, the present invention can include a variety of combinations and/or integrations of the embodiments described herein.

[0017] As described in greater detail below, a proxy cache on a network provides a user interface that enables a network administrator or other user to easily specify and/or modify a set of forwarding rules to control the forwarding of content requests within a cache hierarchy. When the proxy cache receives a request for content from a client and the request produces a cache miss, the proxy cache examines the rules sequentially to determine whether any of the user-defined rules applies to the request. If a rule is found to apply, the proxy cache identifies one or more forwarding destinations from the rule and determines the availability of such destination(s). The proxy cache then selects an available destination, if any, based on the applicable rule, and attempts to

establish a connection to that destination. If a connection is successfully established, the proxy cache forwards the request to the selected destination. If no destination encoded in the rule is available, or if the proxy cache is unable to establish a connection with an available destination from the rule, the proxy cache resumes examining the remaining rules sequentially and repeats the foregoing process.

[0018] As will become more apparent from the description which follows, the described technique provides simple, powerful, and easily-extensible user control over the forwarding of application layer (layer 7 of the ISO/OSI model) requests. With the described technique, the user can clearly and easily see and control the sequence in which the rules will be applied and which rules will be executed under any given circumstances. The user can create complex conditions by listing a sequence of simple rules, since the effect of the rules is cumulative. Backup rules can be created for use when intended forwarding destinations are unavailable. Furthermore, adding additional protocol or condition filters requires no change to the cache hierarchy control unit. New rule filters have no unexpected interactions previously written rules, which simplifies the testing and validation process.

[0019] Figure 1 illustrates a network environment in which a proxy cache according to the present invention may be implemented. As illustrated, a proxy cache 1 configured according to the present invention is connected between a LAN 2 and the Internet 3. A number (N) of clients 4-1 through 4-N are coupled

to the LAN 2. A number (M) of origin servers 5-1 through 5-M are coupled to the Internet 3. The proxy cache 1 forwards requests from the clients 4 for content residing on the origin servers 5 and forwards content and/or other responses from the origin servers 5 to the appropriate clients 4. The proxy cache 1 also caches content from the origin servers 5. It may be assumed that the proxy cache 1 operates within a defined cache hierarchy.

[0020] Note that a proxy cache in accordance with the present invention can be used advantageously in network environments other than that shown in Figure 1. For example, a proxy cache according to present invention need not be used to couple clients on a LAN to the Internet. In other embodiments, one or more other types of networks may be substituted for either the LAN 2 or the Internet 3 in the configuration of Figure 1. Furthermore, a proxy cache may be used in either a forward proxying configuration or a reverse proxying configuration consistently with the present invention.

[0021] Figure 2 illustrates a block diagram of the components of the proxy cache 1, according to one embodiment. The proxy cache 1 includes a cache hierarchy unit 20, a request processing unit 21, a rule engine 22, the rule evaluator 23, a user interface 24, a rule encoder/decoder 25, a network unit 27, and hardware drivers 30. Note that each of these elements can be embodied in either hardware, software, or a combination thereof. For example, each of these modules could be embodied as one or more hardwired integrated circuit (IC) devices, such as application specific integrated circuits (ASICs), gate arrays,

programmable logic devices (PLDs), or the like. To facilitate description, however, it is henceforth assumed in this description that these elements are software modules that are executed by a programmable microprocessor or controller within the proxy cache 1.

[0022] In addition to the foregoing modules, the proxy cache also includes a content cache 26, a rules database 28 containing forwarding rules, and a database 29 of cache hierarchy data identifying the structure of and hosts in the cache hierarchy and indications of the availability (status) of such hosts.

[0023] In one embodiment, the request processing unit 21 is layer 7 (application layer) software. The request processing unit 21 is responsible for receiving content requests from clients, determining whether requested content is cached locally in the content cache 26, and forwarding the requests to appropriate destinations when the requested content is not in the content cache 26 (cache misses). In addition, request processing unit 21 may also be responsible for receiving responses to requests from origin servers and forwarding the responses to the appropriate clients.

[0024] The request processing unit 21 interfaces with network unit 27 and drivers 30. In one embodiment, network unit 27 is conventional network layer (ISO/OSI level 3) software that enables the proxy cache to communicate and other devices (e.g., clients, origin servers, and other caches). In one embodiment, drivers 30 are conventional software for driving hardware for referral devices of the proxy cache, such as disk drives, input/output (I/O) devices, etc. Note that

Figure 2 does not show all of the layers of the software stack of the proxy cache 1 (assuming a software based embodiment); however, any layers not illustrated are not germane to the present invention and are well-within the capabilities of those skilled in the art given this description.

[0025] The rule evaluator 23 is invoked by the rule engine 22 and is responsible for evaluating forwarding rules 28 in response to a request, to determine whether any of the forwarding rules applies to the request. The rule evaluator 23 accomplishes this task by determining whether all of the conditions in the rule are satisfied. A condition in a rule may be a function of a variable, such as client IP address, so that the condition is satisfied when the argument of the condition equals or matches the value of the variable in the request. When a rule is found to apply, the rule evaluator 23 indicates to the rule engine 22 any hosts (forwarding destinations) encoded in that rule. The rule evaluator 23 does this by returning a pointer to the cache hierarchy data 29 which encodes the destination(s), and which allows the rule engine 22 to utilize additional information once an action triggers.

[0026] The cache hierarchy unit 20 maintains knowledge of the cache hierarchy, as indicated by the hierarchy data 29. The cache hierarchy unit is responsible for determining the availability (status) of forwarding destinations (e.g., hosts) in the cache hierarchy, selecting an available forwarding destination encoded in a forwarding rule, and indicating the selected forwarding destinations to the request processing unit 21.

[0027] The rule engine 22 is a subset of the cache hierarchy unit 20 and is primarily responsible for invoking the rule evaluator 23 when necessary. A key aspect of the rule engine 22 is that it can cause rule evaluation to be resumed even after an action triggered, as described further below. In addition, each rule can be associated with arbitrary data structures, and thus, can be independent of the cache hierarchy data.

[0028] The user interface 24 allows the user (e.g., network administrator) to create, modify, and delete forwarding rules, so that the user can take explicit control over how the proxy cache 1 forwards requests. The user interface may be a graphical user interface (GUI), a command line interface, or any other suitable type of user interface. The user interface 24 may be accessed by the user using a separate PC connected to the proxy cache 1 via a network (e.g., via the LAN 2) or via a direct connection. In alternative embodiments, the proxy cache 1 may provide its own I/O devices sufficient to generate a user interface display directly to the user. If the user interface is a GUI, it may be presented to the user in the form of a web page, for example. Users specify rules generally in an if-condition-action format, e.g., "if client-ip = 10.53/16 send single-host A". Each rule is referenced by name and encodes when and where a particular set of destinations should be forwarded requests. By permitting a rule to be disabled, a forwarding scheme may be temporarily altered while preserving the rule contents.

[0029] Figures 3A, 3B and 3C show examples of two web pages that may be

generated by the user interface 24 to allow a user to add, modify or delete forwarding rules, using a mouse, keyboard and/or or other conventional input device(s). Specifically, Figure 3A shows a display screen displaying a set of currently specified forwarding rules. Each rule is represented by a separate line in the "Current Rules" table. The "Rule #" indicates the sequence in which the rule will be evaluated when a request is received. To enable or disable an existing rule, the user checks or unchecks, respectively, the "Enable" checkbox 35 of the appropriate rule.

[0030] The proxy cache may be configured with one or more default forwarding rules, stored in rules database 28. The user may use the user interface 24 to selectively disable any default rules and, if desired, to specify new rules to be used instead. An example of this is shown in Figure 3A, in which the last three rules, "HTTP-to-all", "MMS-to-all" and "RTSP-to-all" are default rules; however, the rule "HTTP-to-all" has been disabled by the user, to be replaced by the user-specified rule named "my new rule".

[0031] To create a new rule, the user may to click on the "New Rule" button 32. To edit an existing rule, the user may click in the "Edit" column 33 of the rule the user wishes to edit. This action causes the display screen of Figures 3B and 3C to be displayed, which is an example of a screen for editing a rule (Figure 3C shows a continuation of the screen of Figure 3B). As shown, the user can specify or alter the sequence in which the rule is evaluated (relative to the other rules) in response to a request by editing the value in the "Rule #" box. Also as shown, the

user can create or change the rule name, check or uncheck the "Rule Enable" checkbox to specify whether the rule is enabled or disabled; check one or more of the "Protocols" check boxes to specifying the protocols to which the rule will apply. In addition, the user can optionally specify a condition under which the rule will be executed. The user can check the "None" checkbox to specify that the rule is unconditional or the "Non-cacheable" checkbox to specify the condition that the request is non-cacheable. Alternatively the user may wish to create a customized condition, such as "Server Host Name equals 'www.yahoo.com'" to define when the rule is to be applied. To create a custom condition, the user checks the "Phrase" checkbox, selects one of the options in pull down menu 36 (e.g., "Server Host Name", "Server IP Address, "Server Domain", "Server Port", "Client IP Address, "Cache IP Address", "Cache Port", "Beginning of URL", "URL"); then selects one of the options in pull down menu 37 (e.g., "equals", "does not equal", or "matches"); and then, enters the appropriate argument (e.g., "www.yahoo.com") in field 38.

[0032] Referring now to Figure 3C, which shows a continuation of the Edit Forwarding Rule screen of Figure 3B, the user may specify in the rule the distribution method for the request. One option is to specify that the request be sent directly. An alternative is that the user can specify "Single Host" and select from pull-down menu 41 one of a set of hosts previously defined in the cache hierarchy. The set of selectable hosts is stored in the cache hierarchy data 29. Alternatively, the user can specify a previously defined firewall cluster, parent

cluster or Internet Cache Protocol (ICP) (RFCs 2186 and 2187).

[0033] In one embodiment the user interface 24 converts user inputs specifying rules into a syntax that has the following format:

"<rulename>" {on | off} if [<protocol>] [and [condition][{= | != | "matches"}] [<argument>]] send <dist_method>

where:

elements that appear in brackets "[ ]" are optional;

the symbol "|" between two terms indicates the terms are alternatives;

the symbol "!=" represents "not equal to";

<rulename> is the distribution rule name;

{on | off} indicates whether the rule is enabled or disabled.

<protocol> is the protocol(s) to which the rule applies (valid inputs may be, for example: "http", "tunnel", "ftp", "gopher", "rtsp", "mms", or "none");

[<conditions>] = additional conditions under which rule applies (conditions can be omitted (unconditional), non-cacheable, or associated with characteristics embedded in the request, such that valid inputs may be, for example, none, "noncacheable" or "acl rules");

<dist_method> is the method and the target of request forwarding (valid inputs may be, for example, "direct" or "single-host <host>" or "icp <hostlist>" or "cluster <hostlist>").

[0034] Forwarding rules (both user-specified rules and default rules) are stored in rules database 28. At run-time, or in response to a configuration change, the

rule encoder/decoder 25 converts the stored rules 28 into a format that is understandable by the rule evaluator 23 and capable of supporting complex statements. Each rule is encoded to include one or more forwarding destinations and communications methods. In one embodiment, a rule encoded by the rule encoder/decoder 25 and stored in rules database 28 has the following format:

<action> <distribution method> [<list of destinations>] [<protocols>] [and] [<condition variable >] [argument]

[0035] Brackets "[ ]" enclose items that are optional. Note, however, that essentially any convenient format can be used. To enable the user to view and edit previously defined rules, the rule encoder/decoder 25 also decodes rules stored in database 28 into a format usable by the user interface 24 to a display the rules to the user for editing.

[0036] In proxy caches of the known prior art, there is no provision for resuming evaluation of rules after an action has been taken or for extending a rule by associating it with an external data structure. In contrast, the rule engine 22 of proxy cache 1 includes the ability to resume rule evaluation even after an action is taken, thus providing a dynamic component to what was previously a static process. The rule evaluator 23 also returns a pointer to the cache hierarchy data 29, as noted, so that the cache hierarchy unit 20 may utilize additional information once an action triggers. The cache hierarchy unit 20 may examine other information external to the rule evaluator 23, such as host availability data in the hierarchy data 29, and the rule engine 22 can subsequently call back into

the rule evaluator 23 to resume evaluation.

[0037] The rule evaluator 23 evaluates rules sequentially, and the effect of rules may be altered by the user at any time by re-arranging their order. When a request containing a matching protocol and/or conditions triggers execution of a rule, the cache hierarchy unit 20 determines whether it believes the destinations encoded by the rule are indeed available. The cache hierarchy unit 20 makes this determination based on status information on each of the hosts of which it has knowledge in the cache hierarchy data 29. If no destination encoded by the rule is available, the rule engine 22 causes rule evaluation to be immediately resumed. Otherwise, an available destination is chosen by the cache hierarchy unit 20 (based on the distribution method) and returned to the request processing unit 21, which then attempts to make a protocol-specific connection to that destination. If the request processing unit 21 is unable to establish a connection with the destination, it asks the cache hierarchy unit 20 to make another choice of destination. This action also can trigger resumption of the rule evaluation process.

[0038] The ability to resume rule evaluation permits the creation of well defined failover strategies specifying primary and secondary paths for traffic. The following two rules provide an example of such a failover strategy:

if client-ip= 10.53/16 send single-host A

if client-ip = 10.53/16 send single-host B

[0039] According to the above two rules, when a request originates on a

specific subnet (10.53/16), then it should always be routed to host "A". This routing may be used for a variety of reasons, such as authentication, content filtering, or bandwidth provisions. However if host "A" is unavailable, then host "B" can serve as a standby backup destination, so that the request can proceed uninterrupted. Re-ordering these rules would reverse the notion of primary and backup with respect to host A and host B.

[0040] Complex conditions can also be created by the user, since the effect of the rules is cumulative. For example, assume that a user desires that any requests not destined for servers X, Y or Z be forwarded to host "A". However, if the requests are destined for X, Y or Z and originate from the client network 10.53/16, the user wants the requests to be forwarded to host "A". On the other hand, if the destination is X, Y or Z and the client network is 10.34/16, then the user wants the request to be sent directly. This forwarding scheme may be specified by the user (via user interface 24) as follows:

if server-name != X Y Z send single-host A

if client-ip = 10.53/16 send single-host A

if client-ip = 10.34/16 send direct

[0041] In one embodiment, rather than complicating the programming syntax by introducing multiple condition variables for each statement, a simple, single-condition syntax is used to achieve the same effect. Since the rule evaluator 23 is responsible for the condition variable definition, the rule engine 22 is not modified when new variables are introduced. This approach makes it very easy

to add new functionality (e.g., new rules) while guaranteeing that previously working software (assuming a software based embodiment) in the proxy cache 1 will not experience any unintended side effects.

[0042] Figures 4A, 4B and 4C collectively form a flow diagram of a request forwarding process that may be executed in the proxy cache 1, according to one embodiment of the present invention. Initially, at block 401 the request processing unit 21 receives a content request from a client and determines whether the requested content is stored in the local cache 26. If the requested content is determined to be in the cache 26 (no cache miss) (block 402), the process ends with block 415, in which the request processing unit 21 retrieves the requested content from the cache 26 and sends the requested content to the requesting client. Of course, before doing so the request processing unit 21 may first apply other conditions or perform other appropriate operations, such as any required AAA and/or determining whether the cached content is still "fresh".

[0043] In the event of a cache miss, however, the process continues from block 403, in which the request processing unit 21 enters information about the request into a data template and then calls the cache hierarchy unit 20, passing to it the filled-in template. The template may have any convenient format and includes various information associated with the request, such as the client IP address, destination server name, uniform resource locator (URL) of the requested content, source port, destination port, protocol, and whether the request is cacheable. In response to receiving the template, the rule engine 22 invokes the

rule evaluator 23 at block 404. Next, at block 405 the rule evaluator 23 evaluates the first rule (as indicated by the user-specified "Rule #") in the rules database 28 that is enabled, to determine whether the rule applies to (should be executed) the request, i.e., the condition in the rule is satisfied (or the rule is unconditional) and the protocol specified in the rule matches that of the request. If the rule being evaluated applies to the request (block 406), the process continues from block 407. Otherwise, the process continues from block 416. At block 416, if all rules have been evaluated for this request, the process ends with block 418, in which the cache hierarchy unit 20 signals the request processing unit 21 to attempt to resolve the request itself. If all rules have not yet been evaluated for this request, however, then the process continues from block 417, in which the rule evaluator 23 evaluates the next rule for applicability to the request, followed by block 406 as described above.

[0044] In block 407 (after the rule is found to be applicable to a request), the cache hierarchy unit 20 determines whether the forwarding destination or destinations specified in the rule are available, based on the host status information in the cache hierarchy data 29. If at least one such destination is available (block 408), the process continues from block 409. Otherwise, the process continues from blocks from 416 and 417 (sequential evaluation of the next enabled rule, if any), as described above. If more than one destination is determined to be available in block 409, then at block 410 the cache hierarchy unit 20 selects one of the available destinations in the rule based on delivery

factors specified in the rule.

[0045] The delivery factors specified in a rule that are used by the cache hierarchy unit 20 to select a destination include at least the specified distribution method (e.g., direct, single-host, etc.). Other examples of possible delivery factors that can be used to select the destination are link bandwidth, destination load, and destination weighting. Link bandwidth may be a measure of the bandwidth which can be supported by a link between the proxy cache 1 and a particular destination. Destination load may be, for example, an indication of whether the current load on a selectable destination exceeds a threshold load value for that destination. This information may be part of the availability data stored in the cache hierarchy data 29 and updated as part of an ongoing process. Destination weighting may be an indication of how the forwarding of requests should be distributed between two or more selectable destinations encoded in a rule. For example, one destination encoded in a rule might be weighted to receive 80 percent of all forwarded requests, while a second destination encoded in the rule is encoded to receive 20 percent of all forwarded requests. Of course, use of the destination weighting may be made subject to other factors, such as the destination load.

[0046] Following block 409, the process continues from block 410. If only one destination from the rule was available, then the process bypasses block 410 and continues from block 411.

[0047] At block 411, the cache hierarchy unit 20 passes to the request

processing unit 21 the destination and a connection timeout period. The timeout period will be used by the request processing unit 21 in attempting to make a connection with the selected destination. The timeout period may be computed by the cache hierarchy unit 20 on a per request basis and may be tailored for the destination selected for the request. For example, part of the availability data in the cache hierarchy data 29 may be information on the response time and/or loading of each host in the cache hierarchy. This information may be obtained by the cache hierarchy unit 20 as part of an ongoing background process of maintaining current status information on all of the hosts in the cache hierarchy. One way in which the expected response time of a host can be determined is to establish a TCP connection with that host from time to time (not necessarily in response to any content request) to determine the amount of time required to complete the connection. Hence, the cache hierarchy unit 20 may compute the timeout period for the selected destination based on the expected response time and/or other data relating to that host.

[0048] Next, at block 412 the request processing unit 21 attempts to make a transfer control protocol (TCP) connection to the selected destination, using the timeout period provided by the cache hierarchy unit 20. If the connection attempt is successful (block 413), the process ends with the request processing unit 21 forwarding the request to the destination at block 414. If the connection attempt was not successful, the process continues from block 419, in which the request processing unit 21 requests an alternate available destination from the

cache hierarchy unit 20. At block 420, if there is another destination encoded in the rule which was determined to be available, then the process loops back to block 411, as described above. If no alternate destination is available, the process loops back to blocks 416 and 417, as described above.

[0049] Figure 5 is a block diagram showing an abstraction of the hardware components of the proxy cache 1. Note that there are many possible implementations represented by this abstraction, which will be readily appreciated by those skilled in the art given this description.

[0050] The illustrated system includes one or more processors 51, i.e. a central processing unit (CPU), read-only memory (ROM) 52, and random access memory (RAM) 53, which may be coupled to each other by a bus system 57 and/or by direct connections. The processor(s) 51 may be, or may include, one or more programmable general-purpose or special-purpose microprocessors, digital signal processors (DSPs), programmable controllers, application specific integrated circuits (ASICs), programmable logic devices (PLDs), or a combination of such devices. The bus system (if any) 57 includes one or more buses or other connections, which may be connected to each other through various bridges, controllers and/or adapters, such as are well-known in the art. For example, the bus system 57 may include a "system bus", which may be connected through one or more adapters to one or more expansion buses, such as a Peripheral Component Interconnect (PCI) bus, HyperTransport or industry standard architecture (ISA) bus, small computer system interface (SCSI) bus,

universal serial bus (USB), or Institute of Electrical and Electronics Engineers (IEEE) standard 1394 bus (sometimes referred to as "Firewire").

[0051] Also coupled to the bus system 57 are one or more mass storage devices 54, a network interface 55, and one or more input/output (I/O) devices 56. Each mass storage device 54 may be, or may include, any one or more devices suitable for storing large volumes of data in a non-volatile manner, such as a magnetic disk or tape, magneto-optical (MO) storage device, or any of various forms of Digital Versatile Disk (DVD) or CD-ROM based storage, or a combination thereof. RAM 53 and/or the mass storage device(s) 54 may be used to implement the content cache 26 (Figure 2).

[0052] The network interface 55 is one or more data communication devices suitable for enabling the processing system to communicate data with remote devices and systems via an external communication link 60. Each such data communication device may be, for example, an Ethernet adapter, a Digital Subscriber Line (DSL) modem, a cable modem, an Integrated Services Digital Network (ISDN) adapter, a satellite transceiver, or the like. Referring again to the embodiment of Figure 1, the network interface 55 is used by the proxy cache 1 to communicate both over the LAN 2 and over the Internet 3. In particular, the network interface 55 is the communications interface by which the proxy cache 1 receives and communicates requests and responses between clients and servers. In addition, the network interface 55 may also be the communications interface by which a user adds, modifies, or deletes forwarding rules used by the proxy

cache 1. Note that while only one external communication link 60 is illustrated, separate physical communication links may be provided for each network connection (e.g., to LAN 2, Internet 3), although that is not necessarily the case.

[0053] Since proxy cache 1 may be accessed by a user via network interface 55, proxy cache 1 does not necessarily require its own I/O devices 56. Nonetheless, such I/O devices may be included in some embodiments and may include, for example, a keyboard or keypad, a display device, and a pointing device (e.g., a mouse, trackball, or touchpad).

[0054] As noted, the above-described processes and techniques (e.g., request processing, rule definition, rule evaluation, etc.) may be implemented at least partially in software. Such software may be part of the operating system of the proxy cache 1. Such software may reside, either entirely or in part, in any of RAM 53, mass storage device(s) 54 and/or ROM 52. Such software may be executed by the processor(s) 51 to carry out the described processes and techniques.

[0055] Thus, a method and apparatus for forwarding requests in a cache hierarchy based on user-defined forwarding rules have been described. Although the present invention has been described with reference to specific exemplary embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of the invention as set forth in the claims. Accordingly, the specification and drawings are to be regarded in an illustrative sense rather than

a restrictive sense.